Open Access Semantic Aware Business Intelligence

Oscar Romero and Alberto Abelló

Universitat Politècnica de Catalunya, BarcelonaTech
oromero@essi.upc.edu,aabello@essi.upc.edu

Summary. The vision of an interconnected and open Web of data is, still, a chimera far from being accomplished. Fortunately, though, one can find several evidences in this direction and despite the technical challenges behind such approach recent advances have shown its feasibility. Semantic-aware formalisms (such as RDF and ontology languages) have been successfully put in practice in approaches such as Linked Data, whereas movements like Open Data have stressed the need of a new open access paradigm to guarantee free access to Web data.
In front of such promising scenario, traditional business intelligence (BI) techniques and methods have been shown not to be appropriate. BI was born to support decision making within the organizations and the data warehouse, the most popular IT construct to support BI, has been typically nurtured with data either owned or accessible within the organization. With the new linked open data paradigm BI systems must meet new requirements such as providing on-demand analysis tasks over any relevant (either internal or external) data source in *right-time*. In this paper we discuss the technical challenges behind such requirements, which we refer to as *exploratory BI*, and envision a new kind of BI system to support this scenario.

Key words: semantic web, business intelligence, data warehousing, ETL, multidimensional modeling, exploratory business intelligence, data modeling, data provisioning

1 Introduction

The Internet empowered the interconnection of different systems and contributed to the bloom of massive and heterogeneous distributed systems that brought new challenges of data integration. The data integration problem [1] aims at providing users with a single unified view of different and interconnected data repositories. This is an old and recurrent topic for the database community and as such it has been thoroughly studied in the past. As discussed in [2], data integration must overcome a number of heterogeneities present in the systems to be interconnected that can be classified in two categories: *system* and *semantic heterogeneities*. On the one hand, system heterogeneities include differences in hardware, operating systems, database management systems (DBMS) and so on.

On the other hand, semantic heterogeneities include differences in the way the real-world is modeled in terms of the database schema.

In the recent past, when the database world was mostly relational, different architectures were developed for data integration, such as federated databases, multidatabase systems and wrappers and mediators [3]. However, semantic heterogeneities are still an open issue for relational systems. Data semantics gathered (as metadata[1] in the database catalog) by relational systems (RDBMS) are not enough as to enable automatic design decisions to overcome semantic heterogeneities and most of the burden to get rid of such heterogeneities still relies on the designer's shoulders.

The arrival of the Internet did nothing but worsen the problem. Massive distributed scalable Web-systems put the traditional relational systems under stress and nowadays relational systems co-exist with non-relational systems, commonly known as NOSQL (to be read Not Only SQL and never as ¬SQL). The first systems that coined the NOSQL term were born on the Web but soon the idea spread to many other areas and currently it claims for specialized database solutions for specific problems. Although NOSQL is mainly a buzzword referring to a way of doing (perfectly captured in the "*one size does not fit all*" motto) rather than new technical solutions, there have been some attempts to classify and find common aspects of these systems. One of the most spread features among NOSQL systems is being *schemaless*. A schemaless database does not have a explicit schema created by the user (e.g., by means of the `CREATE TABLE` statement for RDBMS). Nevertheless, an implicit schema remains. In general, this situation is not desirable at all, as semantics are lost. In terms of the ANSI / SPARC architecture, schemaless databases do not define external and conceptual schemas and query languages must access the DBMS internal data structures, which violates the logical and physical independence principle. Therefore, data is a black-box with no *meaning* for the DBMS and, for example, in some extreme cases such as key-value stores, the value becomes a meaningless chunk of bytes. As consequence, one gains in flexibility but misses even more semantics as metadata gathered by most NOSQL systems is almost non-existent.

As consequence, the current picture is that of massive (independent) systems gathering few metadata, known as *data silos*, which ideally should intercommunicate in order to solve bigger problems. However, data integration becomes even tougher as a wider range of system and semantic heterogeneities must be considered.

Far away from this scenario stands Tim-Berners Lee's vision of an *interconnected* and *open* Web of data [4]. The Semantic Web envisioned by Berners Lee considered *linking* data in such a way a machine could process the links and automatically explore the data without human intervention. Relating data *opens* the data silos and allows navigating, crossing, exploring and analysing data in order to produce unexpected results and relevant (hidden) knowledge. Berners

[1] Metadata, or data about data, keeps track of any relevant information regarding data. For example, a value of 4 means nothing by itself. But if the system knew it refers to the number of children of a certain person as of 2013 it becomes information.

Lee referred to this feature as data fusion [5], and its most popular implementation is by means of *linked open data*. On the one hand, enriching data with machine processable metadata so that machines can *understand* (i.e., manipulate) data is known as linked data [5]. On the other hand, similar to the open source concept, open data describes data that is freely available and can be used as well as republished by everyone without restrictions from copyright or patents [6].

From the database point of view, the linked open data wave demands new systems able to store linked data (thus, semantically rich metadata must be stored together with data) and support the open data initiative (these systems should be easily identified and accessed by anyone). Such systems would provide foundations for more elaborated applications such as mashups, linkeable browsers and semantic-aware search engines. In this paper, we focus on the need for new generation decision support systems built on top of that, which can also be used as foundation for any traditional data fusion application on the Web.

2 Business Intelligence: Past, Present and Future

Decision support systems play a key role in many organizations. These systems provide accurate information (understood as the result of processing, manipulating and organizing data in a way that adds new knowledge to the person or organization receiving it) that leads to better decisions and gives competitive advantages. In the past, the ability of decision makers for foreseeing upcoming trends was crucial for any organization but this largely subjective scenario changed when the world became digital. Actually, any event can be recorded and stored for later analysis, which provides new and objective business perspectives to support managers in the decision making process. Hence, (digital) information is a valuable asset to organizations and gathering, transforming and exploiting such information is nowadays a technological challenge commonly known as *Business Intelligence* (BI). Thus, BI embraces many different disciplines (e.g., from databases to data mining) and solutions meant to support decision making based on (digitally recorded) evidences.

Among all architectural solutions proposed for BI, data warehousing is possibly the most popular one. According to [7], *Data Warehousing is a collection of methods, techniques and tools used to support knowledge workers -senior managers, directors, managers and analysts- to conduct data analyses that help with performing decision making processes and improving information resources.*

Data warehousing mainly focus on decision making and data analysis, and at the same time, these systems abstract technical challenges like data heterogeneity or data sources implementation. This is a key factor in data warehousing in particular, and business intelligence in general. Nowadays, many events can be recorded within organizations but the way each event is stored differs in every organization and it depends on several factors such as relevant attributes for the organization (i.e., their business needs), technology used (i.e., implementation),

analysis task performed (i.e., data relevant for decision making), etc. These systems gather and assemble relevant data available within the organization from various and probably heterogeneous sources in order to produce a single, detailed view of the organization that can be used for enabling better strategic management. In other words, data warehousing overcomes the data integration problem by means of data consolidation.

Although data warehousing ecosystems are highly complex systems, one can say that data warehousing is mature enough (even if data warehousing projects are still far away from being well-controlled). After 20 years, several methods and tools to support the design, deployment and maintenance of such systems have been introduced (e.g., [8, 9, 10, 11]) as long as methodologies and good practices (e.g., [12, 13]). As result, there exist data warehousing experts who have been working and mastering these problem for several years.

Building a data warehouse system consists of designing the data warehouse and creating the ETL (Extract-Transform-Load) processes to populate the data warehouse from the sources. Ideally, the data warehouse schema should subsume any analytical requirement that end-users may pose to the system. Past experience has highlighted three main challenges that complicate designing and deploying data warehousing systems [7, 8]:

1. Like in any other information system, the design process starts by eliciting and gathering the end-user requirements. The burden of such process relies on the DW designer and the end-user does not actively participate in this task. However, it has been shown that IT people (i.e., the DW designer) and non-IT people (i.e., the DW end-user) do not understand each other easily and it is rather common that this step fails to meet the end-user requirements and, in turn, the whole DW ecosystem fails.
2. After gathering the end-user requirements, the designer proceeds to design and create the DW. Next, the ETL processes to populate the DW from the available sources are designed and implemented. In this step, the DW designer is meant to understand the available data sources, identify the data with which the requirements gathered can be meet and construct the ETL flows. This step may take up to 70% of the whole DW project span of time and they entail complex and time-consuming transformations. Accordingly, the update window of a data warehouse (the time it takes to load data into the data warehouse) can be of hours, and executed daily or even weekly. Also, ETL constructs are several times directly implemented at the logical or physical level, which troubles its maintenance.
3. Finally, once the DW ecosystem is running, intuitive and non-technical means to query the DW are mandatory. End-users tend to be non-IT people and require ad hoc formalisms to understand and exploit the data warehouse. For example, one cannot assume that the company CEO masters the SQL language as to query a database on his / her own.

For all these reasons data warehousing projects are traversal and people from different areas must work together in order to produce a flexible, powerful,

and successful system. Ideally, the construction of the DW and ETL designs must undergo several rounds of reconciliation and redesigning, until all business needs –even some that are not described from the beginning of the project– are satisfied. Thus, a DW project is rarely completed. Typically, a DW is an alive ecosystem and thus, maintainability and evolution aspects should be considered too [14].

Beyond its complexity, a successful DW project is a valuable asset for any organization that can effectively exploit the objective evidences stored in its data sources. However, traditional data warehousing techniques have shown not to be adequate for the next generation of BI systems. In terms of the previous section, traditional data warehousing techniques are meant to create independent data silos, whereas current trends claim for interconnecting different data sources (including external sources) and provide a global unified view. This evolution of the data warehousing systems is referred to as DW 2.0, new generation BI and similar concepts [6, 15, 16, 17, 18, 19, 20] that have bloomed recently. According to them, ideally, (i) the end-user should dynamically explore any data source of potential interest (no matter if it is internal or external to the organization) that is available and (ii) any desired analysis task should be made in *right-time* (i.e., according to the user needs, which usually means near real-time). A thorough analysis of these requirements elicit new needs for the next generation BI systems. Specifically:

1. Any desired task in right-time implies:
   - The analysis must be conducted over fresh data and ideally, get rid of the update window concept.
   - The end-user must be able to state his / her analysis requirements in a non-technical language. Ideally, the end-user must state what data want to analyze and from which perspectives without the intervention of the DW designer.
2. Analyze any source means to reconsider how ETL processes should work:
   - Extraction: If any source can be considered, flexible extraction techniques must be able to extract data from structured, semi-structured and non-structured data.
   - Transformation: According to the end-user analytical needs, the extracted data must undergo different transformation rounds to meet the desired quality threshold to be agreed with the end-user.
   - Load: Loading data into the data warehousing can be costly. For this reason, and honouring the right-time requirement, next generation systems may decide not to materialize the consolidated data into a data warehouse but dynamically consume it.

Of course, all these requirements must be put into perspective and they will be rarely met. A graphical representation of traditional and new BI flows is depicted in Figure 1, and a discussion follows.

As shown in Figure 1, data sources can be globally divided into two groups: internal and external data. Internal data is that owned by the company and it traditionally consisted of relational databases and tabular data (e.g., Excel or CVS

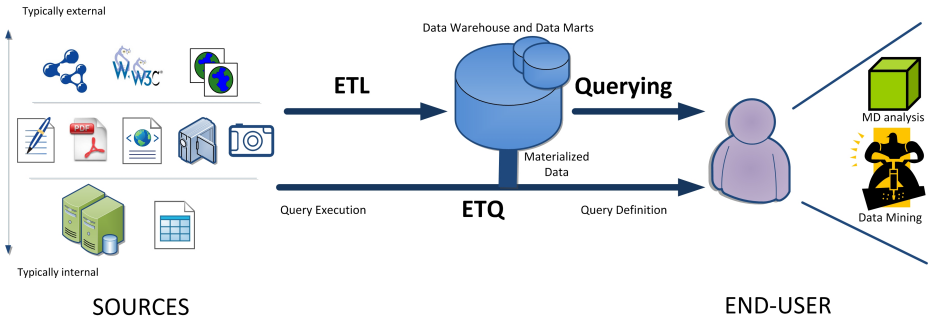

Fig. 1. BI Elements and Data Flows

files). With the time, semi-structured data (such as XML) or non-structured data (e.g., plain text files, pdf files, videos or images) were also incorporated and, nowadays, NOSQL sources (e.g., graph databases or key-value / document stores) may need to be considered. External data mainly refers to open and / or linked data (ideally, open linked data). Open data normally consists of semi-structured data. Its structure typically depends on who delivers the data. For example, governmental institutions[2] usually open tabular data or PDF files, while social networks such as Facebook or Twitter usually deliver XML or JSON files[3]. As special case, external data coming from other organizations after subscribing an agreement may also be considered. This is a typical case in e-science scenarios where research institutes tend to collaborate and share their data. In these cases, an agreement is signed and external access to any kind of source may follow. In any case, a vast amount of heterogeneous data sources may need to be considered.

Next, after undergoing several transformations, the extracted consolidated data might be materialized in a DW. In such scenario, smaller, specific data marts might be modeled. Analytical tools (i.e., data mining, OLAP or query & reporting) are available on top of the DW (or data marts) to allow non-technical end-users to query, navigate and exploit the data. This suits traditional data warehousing techniques. However, next BI systems claim for a more flexible architecture where the end-user can define a query (what data to be analyzed and how) and the system would dynamically decide what sources should be considered, extract data (maybe also from the available DW), do the corresponding transformations and visualize the results. To stress that this approach is query-oriented, we refer to it as ETQ (Extract-Transform-Query). Furthermore, given that the data warehouse could not even exist, it is sometimes meaningless to use the data warehousing term to refer to this new kind of systems, which are better defined as Business Intelligence systems. From here on, we will refer to tradi-

[2] For example: `http://data.gov.uk/` and `http://www.data.gov/`

[3] See `http://www.w3.org/DesignIssues/LinkedData.html` for a detailed description of the 5-stars of linked data.

tional data warehousing systems as traditional BI and next generation systems as exploratory BI. Thus, we stress the fact that in these systems the aim is at *exploring* data sources to perform right-time analytical tasks.

Importantly, note that exploratory BI requires a huge degree of automation in both design and implementation of the query and ETQ tasks. In traditional BI systems the designer is responsible for properly understanding the business requirements, look for the proper data in the sources needed to answer requirements and produce correct data warehouse / ETL models meeting all requirements. Similarly, the designer is responsible for a correct integration of new requirements that may appear. This scenario must change and exploratory BI systems must support the designer as much as possible, and automate the most possible tasks. Of course, the designer (ideally, the end-user) will need to supervise the process and, eventually, accept or disregard different modeling options automatically produced. However, automation requires machine-processable metadata.

2.1 Challenges of Exploratory BI

In the previous section we have just discussed the characteristics of next generation exploratory BI systems. In this section we further elaborate on the IT challenges behind exploratory BI and focus on the technical challenges to overcome in order to (i) *allow the end-user dynamically explore any data source of potential interest (either internal or external to the company)* and (ii) *perform any analysis task in right-time*. More specifically, we divide the technological challenges in three, according to the typical life-cycle of IT systems: requirement engineering, modeling and physical deployment.

Requirement Elicitation and Specification. Exploratory BI systems are goal-oriented (one may consider the analysis needs stated by the end-user as the system goals). Thus, the end-user must express his / her needs without the help of IT people (*requirement elicitation*) and should be internally translated into a machine processable formal specification (*requirement specification*). By needs we either refer to *information needs* (that roughly speaking can be mapped to a query retrieving data) and *quality needs* (some quality criteria that the information retrieved must fulfill and typically expressed in terms of non-functional requirements).

In a truly exploratory system, requirements gathered must be used to lead the next stages (design and implementation of the system) and consequently, the language used to express the user needs (ideally, close to the domain vocabulary used by end-users) must be also *understood* by computers (i.e., it must have precise associated semantics). As later discussed in Section 3, semantic-aware formalisms are the most promising means to describe such *reference domain language*.

All in all, these systems must be *user-centric* and *semantic-aware*. On the one hand, the end-user must express analysis needs using his / her own words. On the other hand, the system must also talk the user language and automatically

process the requirements, which will lead the next two stages. A comprehensive list of challenges related to this area follows:

- End-users must state their *information and quality analysis needs* in terms of a high-level language close to their own vocabulary (i.e., *the reference domain language*).
- The reference domain language must be computer understandable in order to enable the desired automation described in the next two stages.

Automatic Design. From the end-user requirements gathered in the previous stage, exploratory systems must automatically design the system and make this inherent complex task transparent to the end-user. Assuming a data warehousing architecture, it entails to design the integration layer schema (if any) and the ETL (ETQ) processes. More specifically, the system must be able to explore the available (or potential) data sources and, according to the end-user requirements expressed in terms of the reference domain language and some *design quality criteria*, re-arrange the data source schemas in the shape of an integration layer (which can be thought as a view over the data sources) meeting the analysis needs at hand. Once the integration schema is available, the schema transformations identified must be lowered to the instance level and produce the ETL (ETQ) design needed to populate (answer) end-user analysis needs.

Relevantly, note that this entails that the system must be able to understand what data and how is stored at each potential data source. Thus, the system must track the potential data sources and, in order to allow automatic processing, understand what data and how is stored at each source. Thus, it must be able to *map* the data sources to the reference domain language used by the system. Ideally, such map should follow a local-as-view (LAV) approach (similar to that followed by linked data) and thus, concepts in the data sources must point to the reference domain language. In order to automatically consider the end-user quality needs the data sources should also *declare* what quality criteria they might meet.

As consequence, an exploratory system should keep track of potential sources and understand what data and how is stored in the data sources. In other words, to open the black box. To achieve such goal, the data sources schema should be mapped to the reference domain language used by the system. Then, by considering the end-user needs gathered in the previous stage, the exploratory system should be able to design the integration layer and ETL (ETQ) flows needed to answer the analysis needs at hand. A comprehensive list of challenges related to this area follows:

- Potential data sources must be tracked into the system by mapping to the reference domain language both their schematas and the quality criteria they might meet. Ideally, a LAV approach is needed.
- Automatic algorithms for the design of the integration layer and the ETL (ETQ) processes are needed.
- Means to express some design quality criteria to lead the automatic creation of potential designs are mandatory.

Automatic Deployment. The physical deployment of such systems should also be transparent to the end-user. Such deployment should consider the design created in the previous stage and according to some quality needs (either those explicitly stated by the end-user or introduced by the *system administrator*) choose the execution engine to compute the answer to the analysis needs posed to the system. On the one hand, exploratory systems should explore self-tuning systems to the limit and dynamically choose the best storage option as well as auxiliary storage techniques (such as indexing or materialized views) in order to improve performance. On the other hand, an optimizer should be available to optimize ETL (ETQ) executions (similar to the query optimizer of a database). Both issues can drastically benefit from the reference domain language and use the semantic-aware metadata gathered during the whole process in order to determine the best design / execution strategies.

Thus, the system should be able to trigger self-tuning tasks in accordance with the metadata gathered during the whole process, as well as implement an optimizer to improve the ETL (ETQ) internal processes. A comprehensive list of challenges related to this area follows:

- Advanced self-tuning techniques are needed.
- ETL (ETQ) optimizers need to be developed.
- Means to express some quality criteria to guide the system self-tuning and ETL (ETQ) optimization techniques internally carried out are mandatory.

As the reader may note, exploratory systems are user-centered and thus, the ultimate goal of such systems is to allow the end-user state his / her analysis needs (both from the point of view of the information needed and the associated quality metrics associated to it) and accordingly produce the design and physical implementation of the needed constructs in an automatic fashion. As discussed, this must be achieved by means of semantic-aware systems that, by means of a reference domain language, are able to map the end-user requirements and the data source schemas to a common reference domain language. All in all, the database expert role may seem to dilute in this new approach. However, nothing further from the truth. Exploratory systems place the focus on the end-user, who is the real domain expert, and makes him / her play an active role when building the system. The database expert, however, is still essential in this approach. The automatic stages carried out by exploratory systems are responsible for designing and deploying the system. As discussed, some general quality metrics to guide the design and deployment are mandatory. Such guidance must be continuously monitored by a database expert who is expected to react in front of unexpected changes or system misbehaviour (e.g., by changing the design or deployment quality criteria to be met by the automatic created solutions). As consequence, although moved away from the focus, the database expert is still needed to monitor and tune the system.

3 An Introduction to Semantic Web Formalisms

Nowadays, we can find several formalisms to capture semantics in a machine-processable format. Typically, these formalisms come from the Semantic Web (SW), which is aimed at providing the necessary representation languages and tools to express semantic-based metadata. Prior to the SW, there were several efforts to provide metadata formats to the web contents, such as Dublin-Core[4], whose main purpose was to improve information discovery and retrieval. However, these formats were shown very limited mainly due to their poor expressivity and little Web-awareness. As result, the W3C proposed new representation formats, all relying on XML[5], to overcome the limitations of existing metadata formats. The main idea behind these formats is that any concept or instance used for describing a Web object must be referred through a unique resource identifier (URI). Thus, the most basic way to describe an object consists of creating a link to the URI that represents the intended semantics. With the resource description framework (RDF)[6], we can create more complex metadata elements allowing the representation of relationships between descriptors (i.e., triples). Additionally, the RDFS[7] extension allows users to define a schema for RDF descriptions. More expressive semantic descriptions have been also proposed by adopting logic-based frameworks: DAML+OIL[8] and the Ontology Web Language (OWL)[9]. Contrary to RDFS, all these languages rely on description logics, which are tractable subsets of the first order logic (FOL). In this context, metadata is governed by logic axioms over both classes and instances (assertions). Like in RDFS, logic axioms in these formats must be defined over Web-based references (i.e. URIs).

In the next sections we further elaborate in the most popular SW formalisms: RDF and ontology languages.

3.1 RDF(S)

In RDF there are three kinds of elements: resources, literals, and properties. Resources are web objects (entities) that are identified through a URI, literals are atomic values such as strings, dates, numbers, etc., and properties are binary relationships between resources and literals. Properties are also identified through URIs. The basic building block of RDF is the triple: a binary relationship between two resources or between a resource and a literal. For example, consider the following triples:

In this example, the concept (object) eBISS 2013 is represented by the `http://uri-repository/eBISS2013` URI and it is related through the `http:`

[4] `http://dublincore.org/`
[5] `http://www.w3.org/XML/`
[6] http://www.w3.org/RDF/
[7] `http://www.w3.org/TR/rdf-schema/`
[8] `http://www.w3.org/TR/daml+oil-reference/`
[9] `http://www.w3.org/TR/owl2-overview/`

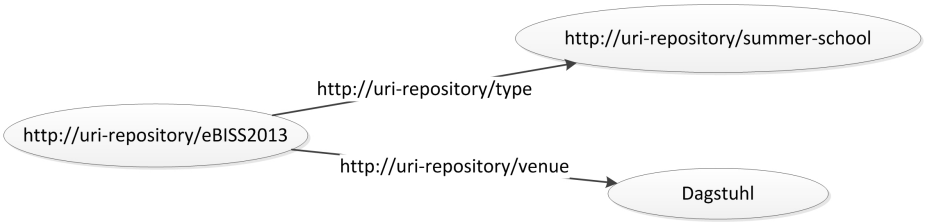

Fig. 2. A graph representing RDF triples

`//uri-repository/type` property to another resource representing the summer school concept `http://uri-repository/summer-school`. Similarly, it is related to the literal *Dagstuhl* by the `http://uri-repository/venue` property. The resulting metadata can be seen as a graph where nodes are resources and literals, and edges are properties connecting them. RDFS extends RDF by allowing triples to be defined over classes and properties. In this way, we can describe the schema that rules our metadata within the same description framework. It is worth mentioning that the semantics of RDFS are based on type systems, similar to object-oriented formalisms and, for example, we can specify *classes*, *subclasses* and typed properties.

3.2 Ontology Languages

Two main families of logic-based languages currently underlie most of the research done in this direction: Description Logics (DL) and datalog-related logics (see [21] and [22], respectively). For example, OWL is founded in DL.

Both, DL and datalog, seek the same objective, but from different points of view. While DL focuses on representing knowledge, datalog is more focused on capturing the instances (and, in this sense, closer to the database field). As discussed in [23], both paradigms can be used to establish ontologies.

On the one hand, DL (or DL-based languages) assume a decentralized approach and information is stored separate from data. Thus, one talks about *terminology* and *instances asserted* (in terms of the terminology). DL also follow the open-world assumption and, accordingly, a DL ontology can have many different interpretations. On the other hand, datalog follows a centralized viewpoint, the closed-world assumption (there is a single interpretation which corresponds to the current database state) and the unique name assumption (two instances differing in their identifier are automatically assumed to be different).

A direct consequence is that DL ontologies are more difficult to model (as "unexpected information" could be inferred from the asserted instances) but they better deal with incomplete data (such as Web data), whereas datalog ontologies are more intuitive for the database community but might not be that interesting for integration cases with missing or partial information.

Modeling in DL and datalog deserve further discussion and basic knowledge on FOL. We address the interested reader to [24] for a discussion on the ex-

pressivity of two popular ontology languages for data modeling: DL-Lite and Datalog±.

All in all, the open-world assumption in DL and the fact that these logics do not follow the unique name assumption suits the essence of Web data (by definition incomplete and where different repositories could identify the same real-world instance by means of different identifiers). From the point of view of BI, DL suits what-if analysis and scenarios with lack of information. In turn, scenarios where the data gathered is known to be complete (to some extent, what is stored in a DW) may be better captured in a datalog ontology.

Need Vs. Feasibility Besides being very expressive, ontology language provide reasoning techniques to infer non-explicit knowledge from already asserted knowledge. Although logic-based languages are very appealing for their semantic-awareness and reasoning features, it is also true that reasoning is known to be computationally hard. Nowadays, it is well established that we must balance the language expressiveness and reasoning services provided according to each scenario.

The main reasoning services provided by DL are concept satisfiability, subsumption and query answering [21]. Concept satisfiability checks if a concept is non-contradictory (regarding the ontology terminology) and it may have, at least, one instance. For example, concept satisfiability (or unsatisfiability) is useful for validating the correctness of the ontology concepts. Subsumption checks if an ontology concept $\mathcal{C}$ is subsumed by another concept $\mathcal{D}$ (i.e., if $\mathcal{D}$ is more general than $\mathcal{C}$). For example, subsumption can be used to identify concept taxonomies and equivalence (if two concepts subsume each other). Finally, query answering finds all the asserted instances that satisfy a concept description and thus, it is extremely useful to pose arbitrary queries over the ontology.

Concept satisfiability and subsumption sit at the terminological level, whereas query answering also deals with instances. Relevantly, very few DL languages (e.g., DL-Lite in [25] and the OWL2 QL profile, based on DL-Lite) properly support query answering which means that, in practice, query answering is prohibitively costly for large data sets, such as those in BI scenarios. Thus, most DL languages are typically used at the terminological level.

Regarding datalog, since terminology and instances are not separated, its reasoning services are query-oriented and its most typical inference is query answering.

4 One Step Towards Exploratory BI

Exploratory BI is, by nature, challenging. However, the current state of the art on BI systems envision a promising future. In this section, we present a functional architecture based on existing approaches towards the creation of a truly exploratory BI. Nonetheless, several challenges remain open but our aim is to show that exploratory BI is no longer a chimera but a feasible challenge.

In the recent past, we have been working on bringing new flexible and powerful BI capabilities to the end-user. Our focus has been set on accessing different data sources (inter / intra organizations / open linked data sources) in a flexible manner: the end-user poses his / her analytical needs and the system is able to produce data cubes on-demand. Ideally, the end-user should state his /her analytical needs and some quality criteria and receive the required data, while the inherent internal complexity of such system should be transparent to him / her. As consequence, the system would be responsible for identifying the data sources, extract and transform the data prior to show it to the user. Performing such tasks on the fly would be extremely costly and therefore, we propose here our vision of a self-tunable architecture that automatically performs optimization tasks to guarantee the feasibility of exploratory BI.

4.1 Narrowing the Focus: Assumptions Made

Before introducing our vision in detail, we need to narrow the focus and properly define what kind of BI systems we do tackle in this architecture. As the reader will note we talk about *cubes*, which is a multidimensional (MD) concept. The multidimensional model was introduced by Kimball [8]. Specifically, multidimensionality is based on the fact / dimension dichotomy. The fact, or subject of analysis is placed in the n-dimensional space produced by the analysis dimensions. We consider a dimension to contain an aggregation hierarchy of levels representing different granularities (or levels of detail) to study data, and a level to contain descriptors (i.e., level attributes). We differentiate between identifier descriptors (univocally identifying each instance of a level) and non-identifier. In turn, a fact contains analysis indicators known as measures (which, in turn, can be regarded as fact attributes). One fact and several dimensions conform a star-schema. Several star-schemas conform a constellation. A specific level of detail for each dimension produces a certain data granularity or data cube, in which place the measures. Thus, one can think of a cube as a query over a star-schema.

Despite its simplicity, the multidimensional schema has been shown to suit analytical tasks [26] and for this reason, we consider the multidimensional model as the *de facto* standard for BI data modeling. For modeling data flows, such as ETL processes, we do not use the multidimensional model (data-oriented) but BPMN (process-oriented). BPMN (Business Process Model and Notation)[10] is an OMG standard that has already been successfully applied to model BI processes in general, and ETL processes in particular [27]. BPMN is a graphical notation that provides constructs to control and manage data flows with enough detail as to be easily translated into an executable flow (for example, using BPEL, Business Process Execution Language).

All in all, the assumptions made in this architecture are as follows:

– A new generation data warehousing system is considered (i.e., we will talk about the integration layer and the ETL/ETQ layer).

[10] See `http://www.bpmn.org/`

– We consider the multidimensional model as the de facto data model for data analysis.
– We consider BPMN as the de facto data model for data workflows.

Obviously, other similar assumptions could be made for alternative solutions.

4.2 Functional Architecture

Next follows a sketched figure of our proposal of an open-access semantic-aware platform for exploratory business intelligence:

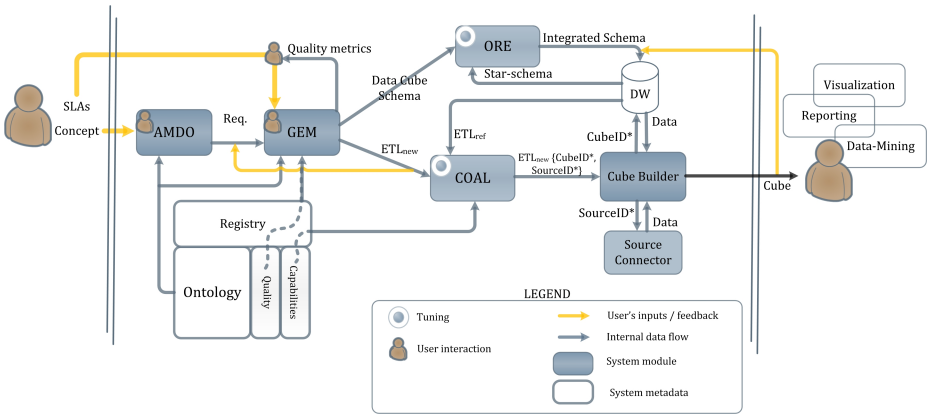

Fig. 3. An open access semantic-aware architecture for business intelligence

First, note that our system is built following two principles previously discussed and motivated in this paper: open data and semantic-aware systems. We say our system is open-access because we aim at providing foundations for exploratory BI on top of freely available and accessible data sources, whereas it must be semantic-aware to enable automation. In our case, the common semantic framework is provided by a reference ontology.

In this system, the user is meant to interact by providing a *seed* or key concept for his / her analysis needs. This concept reaches the first module, AMDO (*Automating Muldimensional Design from Ontologies*) that looks for this concept in the reference ontology. Next, AMDO exploits the knowledge captured in the ontology to propose a list of potential facts, dimensions, measures and descriptors of interest related to the seed concept. This information is shown to the user in a comprehensive way. In short, most relevant concepts (according to some internal rules) are properly ranked and shown to the user, who selects those of his / her interest in a dynamic, interactive manner.

The choices made by the user out of AMDO's suggestions (from now on, the end-user requirement) are forwarded to the GEM module (*Generating ETL and*

Multidimensional Models), which is responsible for producing the data cube design (both the conceptual data cube schema and the conceptual design of ETL flows to provision the cube with data). As consequence, GEM needs to be aware of the end-user non-functional requirements (such as freshness or quality thresholds for the data retrieved) and the candidate sources from where to extract data. Non-functional requirements are expressed in terms of SLAs (service-level agreements), whereas data sources are identified from the registry module. In our system, any data source to be *federated* into the system must be properly registered. In our vision, we should follow a local-as-view (LAV) approach (similar to that followed by linked data) and thus, concepts in the data sources must point to the reference ontology (thus we also assume semantic-aware sources). Accordingly, when registering, the sources must *declare* what ontology concepts they refer to. From now on, we will refer to these ontology concepts as *linked concepts* (as, at least, one source is *linking* them). Furthermore, some quality criteria about the data source (again, as SLAs) must be provided and finally, the registry also must keep trace of the source technical capabilities (such as underlying technology -e.g., relational, triple-store, key-value-, query language, etc.), which will be needed to later query the source. GEM queries the registry to know what sources may provide the required data and the available sources are presented to the end-user together with the quality metrics (extracted from the SLAs) gathered for each of them.

As output, GEM designs a data cube schema and the corresponding ETL flows, as well as SLAs for the query at hand. Note, however, that some non-functional requirements may not be met and at this point the user would be prompted to relax, reinforce or disregard them. Next, the data cube schema is forwarded to the ORE module and the ETL flow to the COAL module. Now, COAL could query the sources and retrieve the needed data according to the ETL flow received. On the contrary, our system performs some optimizations to avoid data shipping (from the sources) whenever possible by materializing some queries in order to improve performance. The ORE module selects relevant pieces of information worth to materialize (e.g., if they are queried regularly) and iteratively consolidates them to produce a complete MD star-schema[11] (potentially, a constellation) subsuming all the cube schemas to be materialized so far. ORE's goal is to create the minimal MD design (e.g., by fusing adjacent facts and /or dimensions, hiding irrelevant concepts, etc.) meeting some tuneable quality criteria. Similarly, COAL is an incremental cost-based method for consolidating individual ETL designs into a unified ETL flow (from now on, the reference ETL) minimizing the execution cost. As result, ORE and COAL generate and maintain a data warehouse.

The system control flow goes to COAL once GEM has generated its outputs. There, COAL is responsible for deciding either to query the data sources or the data warehouse. Whenever possible, the latter will be prioritized. To do

[11] Note we clearly differentiate between a data cube schema and a star-schema. The first one describes the schema of a query, whereas the second one describes a data warehouse schema that can answer many different queries.

so, COAL checks if the new ETL at hand is subsumed by the data warehouse ETL flows. We say the new ETL is subsumed by the reference ETL if after consolidating both of them the output exactly coincides with the reference ETL. In other words, if no changes must be made in order to incorporate the new ETL. In practice, COAL does not look for a full matching but also partial matchings to identify what parts of the new ETL flow can be answered from the data warehouse and what parts must be queried from the sources (in the figure, this has been represented as a set of *cubeIDs* and *sourceIDs*). Intuitively, one may say that COAL tries to *rewrite* the new ETL in terms of the already available ETL flows.

Finally, the cube builder module can be thought as a query executor. It gathers the data retrieved from the sources and the data warehouse and, according to the ETL conceptual schema produced by GEM, builds the cube. If a source must be queried, the source connector triggers a query to retrieve the data according to the registered data source capabilities and capability-based optimization techniques [28]. Note the data sources are black-boxes for our system and, beyond ORE and COAL, we pass the responsibility for optimizing the query to the underlying system. To some extent, one may say ORE and COAL are responsible for the global optimization, whereas data sources are responsible for the local optimization of queries.

Once the cube has been shown to the user, the system (or the user) might decide to materialize it. Then ORE and COAL come into action to integrate the data cube at hand into the data warehouse. Obviously, any visualization or analytical tool might be used to further analyze the produced data cube from the end-user point of view.

Last, but not least, note some relevant features of our approach. According to Figure 1, our ultimate goal essentially corresponds to an ETQ scenario, but the internal optimization techniques enable other data flows besides ETQ. Specifically, COAL and ORE correspond to the ETL arrow and the data warehouse repository respectively, whereas the ETL rewriting technique (i.e., the *CubeID* list produced by COAL) corresponds to the materialized data arrow. From the point of view of use cases, the user is meant to interact with AMDO and GEM, whereas the system administrator (the former DW designer) is meant to interact with ORE and COAL (for example, tuning the internal metrics used by ORE and COAL to consolidate the designs). Furthermore, both ORE and COAL generate and store valuable metadata of interest for the end user. For example, the data warehouse MD schema and ETL processes can be retrieved and visualized. This feature has many applications and, for example, the ETL flows can be used to tackle traceability and show what sources were considered, what transformations applied and how it was combined and visualized as a data cube, whereas the star-schemas can be used to tackle collaborative BI and query recommendation (based on past evidences).

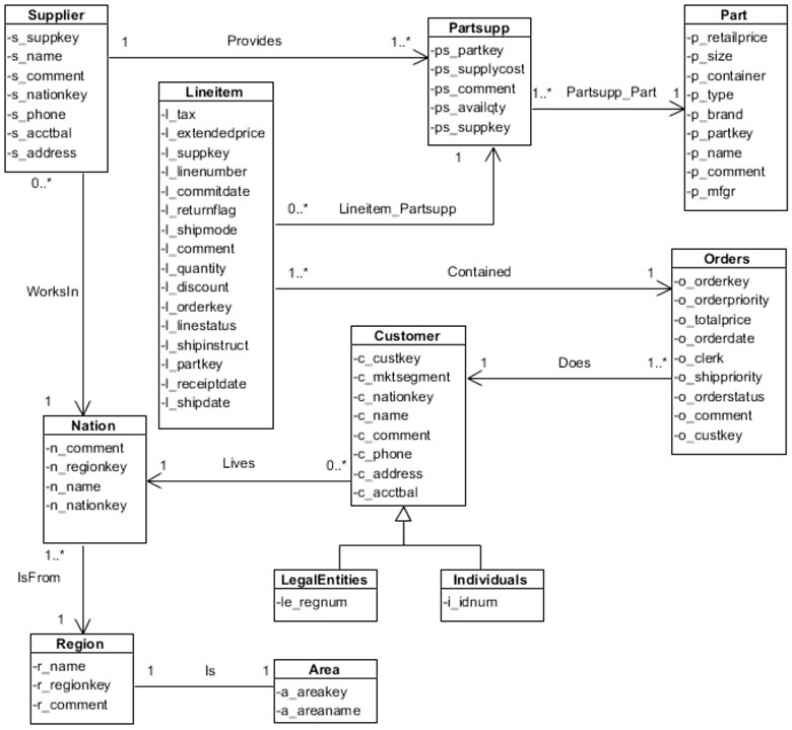

Fig. 4. TPC-H Diagrammatic Representation

4.3 A (Toy) Usage Example

Prior to detailing each of the modules contained in our system (see next sections), we introduce an example to show how such a system would work from the point of view of the end-user. For this example we will consider the TPC-H benchmark [29]. In our approach, we need an ontology (our common semantic framework) whose diagrammatic representation is depicted in Figure 4. Suppose now an end-user interested in *orders lineitems*. It may start dropping a query by writing the word *lineitem* in the GUI. Immediately, AMDO looks for this word in the system reference ontology and proposes a set of dimensions, facts and measures of potential interest. For example, let assume the output shown in Table 1 (in brackets, the relevance computed by AMDO).

Usually, AMDO will propose different facts (not only one) ranked by relevance (see Section 5.1 for further details) and for each fact it will propose different measures and dimensions. Similarly to most BI dashboards, the user may drag and drop the concepts to build the desired cube schema. For example, suppose the user chooses *extendedPrice* and *discount* as measures and the *customer nation* as single dimension. At this point, AMDO will allow the user to add slicers (e.g., *nation = 'Serbia'*) and specify derivation functions (e.g., *price = extendedPrice*(1-discount)*) and aggregation functions (e.g., compute

Proposed fact: *lineItem* (100%)

Measures	Dimensions
extendedPrice (100%)	orderdate (93%)
quantity (95%)	shipdate (90%)
discount (80%)	nation (of the customer) (87%)
...	supplier (83%)
	...

Table 1. An Example of AMDO's outputs

the *average price*). Once done, GEM creates the data cube schema. For example, consider the example depicted in Figure 5. GEM identifies the ontology subset (bolded in colours) needed to satisfy the end-user requirement forwarded from AMDO (in words, *the average price paid* (measure) *by Serbian customers* (dimension)). For further information on GEM see Section 5.2 but note that additional ontology concepts (e.g., *orders* in our running example, which is bolded in green), not chosen by the end-user, may be needed to properly relate the concepts at hand (bolded in orange) and produce a single data cube. Once the schema has been created, GEM looks for those registered sources *linked* to the ontology concepts participating in the cube. These sources are presented to the user together with the data source quality evidences registered as SLAs. Suppose now three sources: A, B and C, but B presents some quality issues: its servers are frequently down and data provided is generated by a small community that cannot guarantee its correctness. For this reason, the user decides not to consider B and proceed with the other two sources. Next, GEM designs the needed ETL processes to provision the data cube schema with data from the selected sources.

Next, COAL is launched to check what parts of the new ETL flow can be rewritten in terms of the data warehouse ETL flows and what others need to query the data sources. Suppose we need to query a source. Then, according to the metadata regarding A and C it asks the source connector to wrap a query to obtain the needed data. For example, if A is a relational database that contains a table `M(orderkey, partkey, suppkey, extendedPrice, discount, ...)` (like the one provided in the TPC-H schema for *lineitem*) and we want to obtain the measures needed from it, it would trigger an SQL query such as `SELECT orderkey, partkey, suppkey, extendedPrice, discount FROM M` (where the set {`orderkey, partkey, suppkey`} is considered to be the table primary key). Data gathered from the data warehouse and the data sources is properly assembled according to the ETL flow created by GEM.

Once the data cube is ready any visualization or analytical tool can be used to show it to the user. If the user eventually decides to integrate this data cube into the data warehouse, our system would launch first ORE and then COAL to perform the needed evolution tasks in the data warehouse (further details in the next section).

Finally, suppose now that we decided to materialize the current query and, in the future, the user is interested in the *discount* (measure) obtained for each

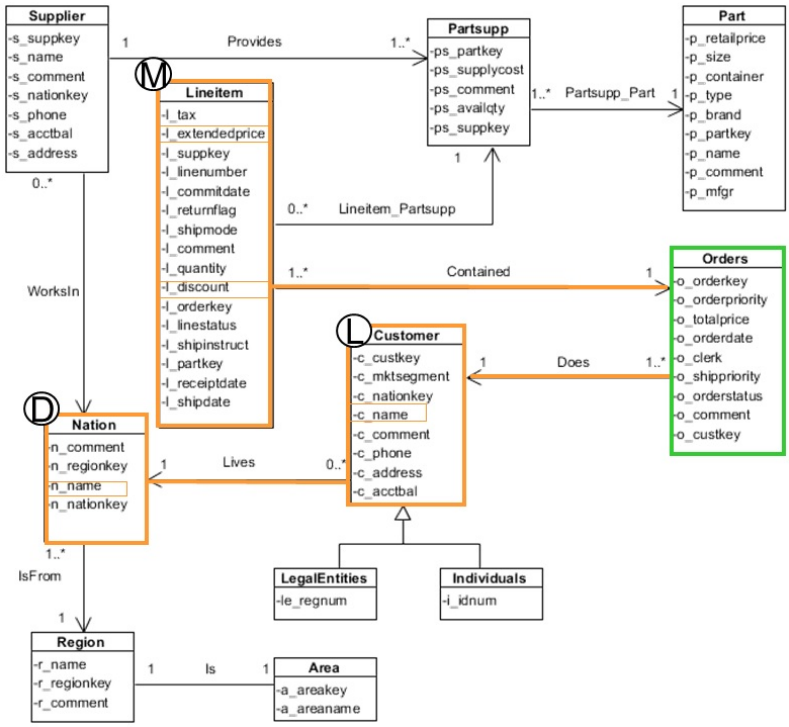

Fig. 5. GEM example

customer (dimension). Clearly, this query can be rewritten in terms of the inputs needed for the query discussed above. Thus, the COAL module will use equivalence rules to rearrange the ETL operations in both flows (always preserving the semantics) and maximize the overlapping area. In this case, the new ETL is completely subsumed by the reference one and therefore, COAL will find a full match (i.e., complete overlapping). See Section 5.4 for further details.

5 An Open-Access Semantic-Aware System

Our system sets a common semantic framework by means of an ontology. Note that, up to now, we have assumed a single reference ontology. However, in practice, several ontologies may co-exist. Ontology matching techniques can be used to combine such ontologies and, in the end, the more inter and intra-relationships captured, the better. Regarding the sources, by now, we only assume semantic-aware repositories (e.g., linked data). Nevertheless, it might be possible to wrap other sources with additional semantics and enable their integration into our system. For example, tools like *Triplify*[12] can be used to semi-automate such

[12] See `http://triplify.org/Overview`

task. This opens the door for a polyglot system consisting of heterogeneous data sources.

In the rest of this section we will focus on the main modules of our proposal (i.e., AMDO, GEM, ORE and COAL) and we present them in more detail.

5.1 The AMDO Module

AMDO [30] looks for ontology concepts that together with the seed concept (provided by the end-user) may produce meaningful data cubes. In practice, it looks for concepts likely to play a valid multidimensional role (with regard to the seed concept). Specifically, dimensions arrange the multidimensional space where the fact of study is depicted. Each instance of data is identified (i.e., placed in the multidimensional space) by a point in each of its analysis dimensions. Conceptually, it implies that a fact must be related to each analysis dimension (and by extension, to any dimensional concept) by a many-to-one relationship. That is, every instance of the fact is related to, at least and at most, one instance of an analysis dimension, and every dimension instance may be related to many instances of the fact.

AMDO looks for potential facts, dimensions, measures and descriptors by means of *topological patterns* guaranteeing the MD constraints above discussed. Importantly, AMDO does not perform a blind search but a guided search from the seed concept. Furthermore, some internal metrics are used to rank concepts found. For example, a fact containing many measures and dimensions, closeness to the seed concept, etc.

Internally, AMDO exploits standard reasoning services to compute the topological patterns.

5.2 The GEM Module

GEM [31] receives the facts, dimensions, measures and descriptors identified by AMDO, which we will refer to as the input requirement from now on.

The process of creating the MD and ETL designs for the input requirement is a semi-automatic process comprising four main stages (see Figure 6). The outcome of each stage is validated and then, either it is propagated to the next stage or undergoes a correction process. The correction process may be done automatically, it may suggest changes, or it may require user feedback.

Stage 1: Requirement verification. First, the system checks if there is a mismatch among the input requirement and the ontology linked concepts. For each concept in the input requirement, GEM checks if, at least, there is a source linked to it. In case of mismatch, it may suggest relaxation of the requirement or alternatives (e.g., choosing subclasses).

Stage 2: Requirement completion. After mapping the input requirement onto the ontology and verifying it, the system complements it with needed additional information. This stage identifies *intermediate concepts* that are not explicitly stated in the business requirement, but are needed in order to retrieve

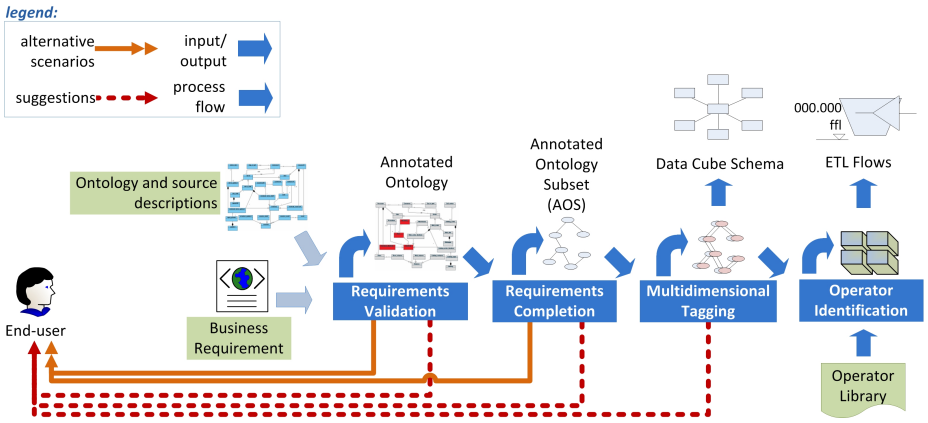

Fig. 6. GEM in a Nutshell

data. Intuitively, it identifies all the ontology concepts needed to eventually build a *query* (or access plan) to answer the requirement. User feedback is welcomed for ensuring correctness and compliance to the end-user needs in case of ambiguity.

Stage 3: Multidimensional verification. Next, we look for a MD interpretation of the ontology subset identified in the previous stage. To do so, we check the MD integrity constraints and verify the correctness of the requirement according to MD design principles. Hence, we check two issues: (i) whether the factual data is arranged in a MD space (i.e., it forms a data cube and thus, if each instance of factual data is identified by a point in each of the analysis dimensions [32]); and (ii) whether data summarization performed is correct by examining whether the following conditions hold [33]: (a) *disjointness* (the sets of objects to be aggregated must be disjoint); (b) *completeness* (the union of subsets must constitute the entire set); and (c) *compatibility* of the dimension, the type of measure being aggregated and the aggregation function.

Stage 4: Operator identification. The ETL operations are identified in three phases. First, we use the annotations generated by the previous steps for extracting schema modification operations. Then, the cubes are built. And finally, we complement the design with additional information that might be found in the sources and with typical ETL operations such as surrogate key and slowly changing dimensions. Similar to Stage 1, once the sources have been identified, this step looks for mismatches between the end-user and data source SLAs and may suggest alternatives or relaxation of some non-functional requirements.

5.3 The ORE Module

ORE [34] is responsible for integrating new data cube schemas into the data warehouse. It comprises four stages, namely *matching facts*, *matching dimensions*, *complementing the MD design*, and *integration* (see Figure 7). The first three stages gradually match different MD concepts and explore new design alternatives. The last stage considers these matchings and designer's feedback to

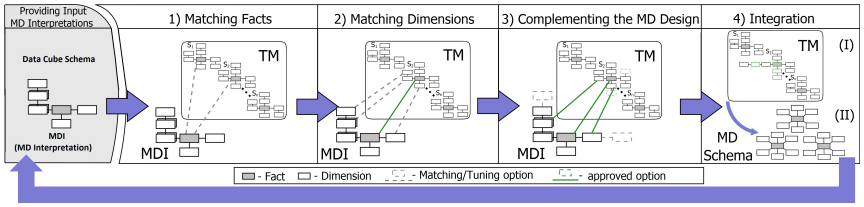

Fig. 7. ORE in a Nutshell

generate the final MD schema that accommodates a new information requirement.

In all stages, we keep and maintain a structure, namely *traceability metadata* (TM), for systematically tracing everything we know about the MD design integrated so far, like candidate improvements and alternatives. With TM, we avoid overloading the produced MD schema itself.

Stage 1: Matching facts. We first search for different possibilities of how to incorporate the data cube schema at hand (i.e., the MD interpretation produced by GEM) to TM. The matching between factual concepts is considered –i.e., the system searches the fact(s) of TM producing an equivalent set of points in the MD space– as the one in the given MD interpretation. Different possibilities to match the factual concepts results with the appropriate sets of integration operations. The costs of these integration possibilities are weighted and prioritized.

Stage 2: Matching dimensions. After matching facts, we then conform the dimensions of the paired facts. Different matchings between levels are considered (i.e., "=", "1 - 1", "1 - *" and "* - 1") and thus, the different valid conformation possibilities are obtained. With each possibility, a different set of integration operations for conforming these dimensions is considered and weighted.

Stage 3: Complementing the MD Design. We further explore the reference ontology and search for new analytical perspectives related to the new concepts. Different options may be identified to extend the current schema with new levels, descriptors, and measures). The user is then asked to (dis)approve the integration of the discovered concepts into the final MD schema.

Stage 4: Integration. The MD schema is finally obtained in two phases of this stage. First, possible groupings of the adjacent concepts containing equivalent MD knowledge is identified to minimize the MD design. Finally, the final MD schema is produced by folding and collapsing grouped concepts to capture only the minimal information relevant to the user. Nevertheless, the complete TM is still preserved in the background to assists further integration steps (e.g., to handle future evolution events).

Example: The general idea behind ORE can be visualized in the example depicted in Figure 8. This figure shows two outputs produced by GEM for two given requirements (IR1 and IR2) over the TPC-H example. As output, ORE will produce an integrated schema meeting the two cube-schemas of the requirements at hand.

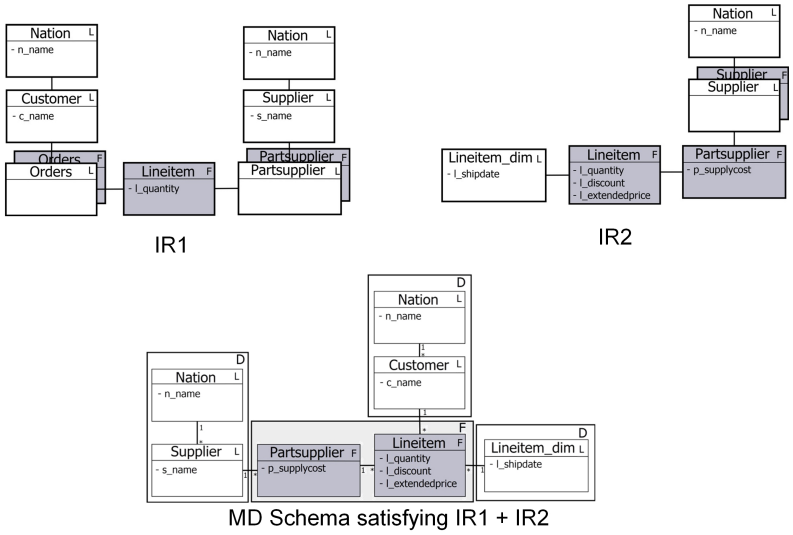

Fig. 8. ORE example

5.4 COAL

COAL [35] is responsible for integrating the new ETL flow at hand with the data warehouse ETL flows. Typically, an ETL design is modeled as a directed acyclic graph. The nodes of the graph are data stores and operations, and the graph edges represent the data flow among the nodes. Intuitively, for consolidating two ETL designs, a referent G_1 and a new G_2 designs, we need to identify the maximal overlapping area in G_1 and G_2. However, this is not a typical graph-matching problem, as the MD interpretation of the ETL flow must be preserved. Therefore, we proceed as follows. First, we identify the common source nodes between G_1 and G_2. For each source node, we consider all paths up to a target node and search for common operations in both designs. In these paths, we search for common operations that could be consolidated into a single operation in the resulting design.

Deciding what operations can be consolidated and how is not an easy task. If two operations, each placed in a different design, can be matched, then we have a *full match* (e.g., the very same or equivalent operations). If two operations, one in the reference design and the other in the new design, partially overlap, then we have a *partial match* (e.g., one operation subsuming the other). To detect the maximum number of full and partial matchings, we should also look at the operations performed before the ones considered for the matching; i.e., COAL must consider restructuring both designs by moving operations to maximize the number of overlapping operations. Restructuring the ETL designs must be performed by guaranteeing the same semantics as result and this is achieved by means of a set of predefined equivalence rules between operations (e.g., selections can always be pushed down a join operation, but a selection

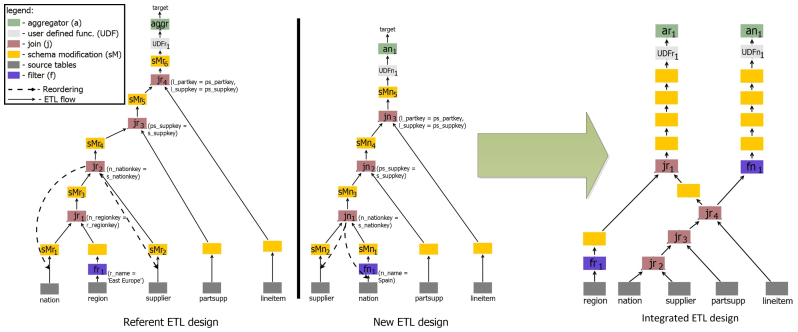

Fig. 9. COAL example

cannot be pushed down a projection if the selection attribute coincides with the projected attributed).

Example: The general idea behind COAL can be visualized in the example depicted in Figure 9. There, a new and a reference ETL flow are presented. In general, these two designs may have a number of common operations. COAL's internal algorithms look for a minimal resulting integrated ETL such that common operations are executed once and the semantics of both flows are preserved by means of the above discussed equivalence rules. What minimal exactly means depends on the quality criteria determined by the system administrator.

5.5 Open Questions

In our functional architecture we have shown the feasibility of some of the challenges behind exploratory BI (see Section 2.1). However, several challenges still remain open for exploratory BI and deserve further attention. More specifically, the most important ones can be summarized as follows:

- **Integration of schemas:** In practice, it is unfeasible to assume that a single common semantic framework (where any potential data source of interest is mapped) does exist. Indeed, several reference languages may co-exist and automatic mappings should be discovered (ideally, by means of reasoning).
- **LAV vs. GAV:** Clearly, a LAV approach is desired for mapping the data sources to the semantic framework but nowadays most approaches follow a GAV approach (i.e., the mappings are in the ontology concepts). Further research on semantic-aware formalisms to allow LAV is desirable.
- **Reasoning:** Tightly related to the previous item, computationally feasible reasoning techniques are needed in order to infer knowledge not explicitly stated in the reference layer. In our work, we used the DL-Lite family [25], which provides a good trade-off between expressivity (similar to that of UML) and computational complexity. However, only basic inference algorithms (such as subsumption) were feasible in practice. Others, like computing the transitive functional closure, needed ad-hoc algorithms to be computed. Thus, creating reasoning facilities over DL / datalog families especially designed to capture

the multidimensional model is a must. In this sense, how to exploit parallelism and benefit from distributed computation when computing reasoning is also an open challenge.
- **ETL Operators:** The ETL flows automatically generated in our approach mainly consider the relational algebra and a bunch of additional operations (create surrogates, dictionary look-ups, etc.). Ideally, any transformation should be able to be specified and automatically considered by our framework in an automatic way (right now, the ETLs produced need to be manually enriched).
- **Non-Functional Requirements:** How to specify non-functional requirements in a machine readable format and include them all over the process is still also an open challenge. Traditionally, non-functional requirements are considered and the database is correspondingly tuned by database administrators.

6 Semantic Aware Business Intelligence: State of the Art

In the recent past there has been a bloom of new techniques and methods for BI relying on semantic-aware formalisms. Semantic-aware data warehouses are nowadays hot topics of research (e.g., among many others [36, 37, 38, 39]). These works can be understood as the cog wheels forming the exploratory BI machine and they span from requirement engineering, conceptual design to physical design in many and disparate areas. Surveying all these works is completely unfeasible and, for this reason, in this section, we will focus on those approaches presenting similar systems to what we have called exploratory BI (i.e., the big picture) and how they propose to combine different techniques to produce similar systems. Indeed, in the literature we can find several equivalent or similar terms to exploratory BI. For example, "live BI" [15], "on-demand BI" [16], "ad-hoc BI" [17], "open BI" [18], "situational BI" [19], or "fusion cubes" in [20].

[18] presents a platform to analyze linked open data and, similar to our approach, they assume semantic-aware sources. Data modeling and provisioning are achieved by means of a traditional data warehouse architecture (which is loaded with data from the sources) following model-driven techniques and their focus is on providing advanced support to non-technical users to trigger data mining algorithms over the gathered data. To support non-expert data mining users, a knowledge base conforming quality criteria of the sources is created and used as main source to recommend the user data explorations.

[16] presents *on-demand* BI services and adopt models related to Software-as-a-Service (SaaS) as technical solution. Their approach consists of a multi-layered architecture to support different business intelligence needs, namely: the designer tasks (designing the environment, perform data integration and manage metadata) based on a model-driven approach and the end-user analytical needs. They add a third tier of services related to security (which includes authorities, roles, users, groups and grants). From the point of view of BI, this solution embeds and supports traditional BI into a SaaS architecture.

[17] focuses on the technical challenges of *ad-hoc* BI, namely, a global business data model, data source integration and enrichment (in which they distinguish between business configuration at design time, and data provisioning at run time) and finally, support for ad-hoc (self-oriented) and collaborative BI.

[15] presents a unified data management and analytics platform for *live* BI. This paper presents a flexible architecture that allows to specify and define ETL flows from highly heterogeneous sources. They introduce the concept of extraction operators so that unstructured or semi-structured data can be extracted and integrated with structured data. In this paper, the data flow is defined in terms of a pipeline transforming input streams into analytics results. It is presented in terms of an event discovery stage (based on historical evidences) from the input stream and a second stage of further analysis of the events to detect and predict non-explicit patterns. Special emphasis is put on the need of a powerful physical design and optimizer that could deal with heterogeneous and mixed data flows.

[19] presents a platform to correlate data from an organization data warehouse with external sources. This platform is database-inspired and enables traditional BI queries (ad-hoc and aggregate queries) over cloud architectures. Their approach consists of a common algebraic core to describe, plan, optimize and execute queries, and similar to the previous approach, they focus on unstructured sources and how to extract and integrate data from them. Also, an optimizer and a parallel executor engine are introduced.

Finally, [20] envisions a highly heterogeneous BI system where a query is formulated, then relevant sources are discovered and selected and data provisioning and integration flows are triggered before presenting the resulting data to the user. An abstract architecture is also presented and data from external and internal sources is ETL-ed into *stationary* (i.e., pre-defined cubes) and *fusion cubes* (similar to the ETQ term in this paper) before querying them. Different from the rest of approaches, this is a visionary paper presenting future research trends and priorities for data modeling and provisioning for BI processes but no specific solution is discussed.

All in all, these approaches present a similar ultimate goal and some common trends can be identified. For example, they agree on the need to make transparent to the end-user all the technical complexity behind advanced BI solutions and provide flexible and intuitive conceptual formalisms in order to allow end-users specify and design their queries. These solutions mainly differ though in the assumptions made. We can identify a key assumption that conditions the solution presented: loosely coupled - tightly coupled data sources. In this aspect, we can find a very wide range of solutions but the consequences are clear: the more tight the data sources are the better internal optimizations can be made.

Regarding our approach, we present an *instance* of the abstract architecture envisioned in [20] and use a reference ontology to relate loosely coupled sources and, at the same time, enable some global optimizations. Furthermore, special emphasis is put on automatic MD discovery, design and deployment.

7 Conclusions

In this paper we have motivated the need for new generation BI systems and coined the exploratory BI term, which should enable end-users to trigger right-time analytical tasks over disparate and heterogeneous sources. The challenges behind exploratory BI are manifold, but we have focused on two key aspects: open-access (following the open data movement) and semantic-aware repositories to enable automation.

As a proof of concept, we have sketched the architecture of an open-access semantic-aware system to support exploratory BI and highlighted the main areas of research for enabling exploratory BI.

References

1. Lenzerini, M.: Data integration: A theoretical perspective. In Popa, L., Abiteboul, S., Kolaitis, P.G., eds.: PODS, ACM (2002) 233–246
2. Bukhres, O.A., Elmagarmid, A.K., eds.: Object-Oriented Multidatabase Systems: A Solution for Advanced Applications. Prentice-Hall (1996)
3. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, Third Edition. Springer (2011)
4. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems **21**(3) (2006) 96–101
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. **5**(3) (2009) 1–22
6. Eberius, J., Thiele, M., Braunschweig, K., Lehner, W.: Drillbeyond: Enabling business analysts to explore the web of open data. PVLDB **5**(12) (2012) 1978–1981
7. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. 1 edn. McGraw-Hill, Inc., New York, NY, USA (2009)
8. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses. John Wiley & Sons, Inc. (1998)
9. Giorgini, P., Rizzi, S., Garzetti, M.: Grand: A goal-oriented approach to requirement analysis in data warehouses. Decision Support Systems **45**(1) (2008) 4–21
10. Mazon, J.N., Trujillo, J., Lechtenborger, J.: Reconciling Requirement-Driven Data Warehouses with Data Sources Via Multidimensional Normal Forms. Data & Knowledge Engineering **23**(3) (2007) 725–751
11. Jensen, C.S., Pedersen, T.B., Thomsen, C.: Multidimensional Databases and Data Warehousing. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2010)
12. Winter, R., Strauch, B.: A Method for Demand-Driven Information Requirements Analysis in DW Projects. In: Proc. of 36th Annual Hawaii Int. Conf. on System Sciences, IEEE (2003) 231–239
13. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern software engineering methodologies meet data warehouse design: 4wd. In: Proceedings of DaWaK 2011. Volume 6862 of Lecture Notes in Computer Science., Springer (2011) 66–79
14. Wrembel, R.: A survey of managing the evolution of data warehouses. IJDWM **5**(2) (2009) 24–56

15. Castellanos, M., Dayal, U., Hsu, M.: Live business intelligence for the real-time enterprise. In: From Active Data Management to Event-Based Systems and More. Volume 6462 of Lecture Notes in Computer Science., Springer (2010) 325–336
16. Essaidi, M.: ODBIS: towards a platform for on-demand business intelligence services. In: EDBT/ICDT Workshops. ACM Int. Conf. Proceeding, ACM (2010)
17. Berthold, H., Rösch, P., Zöller, S., Wortmann, F., Carenini, A., Campbell, S., Bisson, P., Strohmaier, F.: An architecture for ad-hoc and collaborative business intelligence. In: EDBT/ICDT Workshops. (2010)
18. Mazón, J.N., Zubcoff, J.J., Garrigós, I., Espinosa, R., Rodríguez, R.: Open business intelligence: on the importance of data quality awareness in user-friendly data mining. In: EDBT/ICDT Workshops, ACM (2012) 144–147
19. Löser, A., Hueske, F., Markl, V.: Situational business intelligence. In Castellanos, M., Dayal, U., Sellis, T., eds.: Business Intelligence for the Real-Time Enterprise. Volume 27 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 1–11
20. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J.N., Naumann, F., Pedersen, T.B., Rizzi, S., Trujillo, J., Vassiliadis, P., Vossen, G.: Fusion cubes: Towards self-service business intelligence. Int. J. on Data Warehousing and Mining **9**(2) (2013)
21. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: Description Logic Handbook, Cambridge University Press (2003)
22. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering **1**(1) (1989) 146–166
23. Patel-Schneider, P.F., Horrocks, I.: Position paper: a comparison of two modelling paradigms in the semantic web. In: 15th international conference on World Wide Web (WWW), ACM (2006) 3–12
24. Calì, A., Gottlob, G., Lukasiewicz, T.: Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In Fagin, R., ed.: ICDT. Volume 361 of ACM International Conference Proceeding Series., ACM (2009) 14–30
25. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite Family and Relations. J. of Artificial Intelligence Research **36** (2009) 1–69
26. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Proceedings of DaWaK 2011. Volume 6862 of Lecture Notes in Computer Science., Springer (2011) 66–79
27. Akkaoui, Z.E., Mazón, J.N., Vaisman, A.A., Zimányi, E.: Bpmn-based conceptual modeling of etl processes. In: Proceedings of DaWaK 2012. Volume 7448 of Lecture Notes in Computer Science., Springer (2012) 65–80
28. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Pearson Prentice Hall (2008)
29. TPC: TPC-H specification. Available at: http://www.tpc.org/tpch/ (2012)
30. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data Knowl. Eng. **69**(11) (2010) 1138–1157
31. Romero, O., Simitsis, A., Abelló, A.: GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In: Proceedings of DaWaK 2011. Volume 6862 of Lecture Notes in Computer Science., Springer (2011) 66–79
32. Mazón, J., Lechtenbörger, J., Trujillo, J.: A Survey on Summarizability Issues in Multidimensional Modeling. DKE (2009) 1452–1469

33. Lenz, H., Shoshani, A.: Summarizability in OLAP and Statistical Data Bases. In: SSDBM. (1997) 132–143
34. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: ORE: an iterative approach to the design and evolution of multi-dimensional schemas. In Song, I.Y., Golfarelli, M., eds.: DOLAP, ACM (2012) 1–8
35. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Integrating etl processes from information requirements. In: Proceedings of DaWaK 2012. Volume 7448 of Lecture Notes in Computer Science., Springer (2012) 65–80
36. Pérez, J.M., Llavori, R.B., Aramburu, M.J., Pedersen, T.B.: Integrating data warehouses with web data: A survey. IEEE Trans. Knowl. Data Eng. **20**(7) (2008) 940–955
37. Nebot, V., Llavori, R.B.: Building data warehouses with semantic web data. Decision Support Systems **52**(4) (2012) 853–868
38. Spaccapietra, S., ed.: Journal on Data Semantics XII. Volume 5480 of Lecture Notes in Computer Science., Springer (2009)
39. Gallinucci, E., Golfarelli, M., Rizzi, S.: Meta-stars: multidimensional modeling for social business intelligence. In: Proceedings of DOLAP 2013, ACM (2013) 11–18